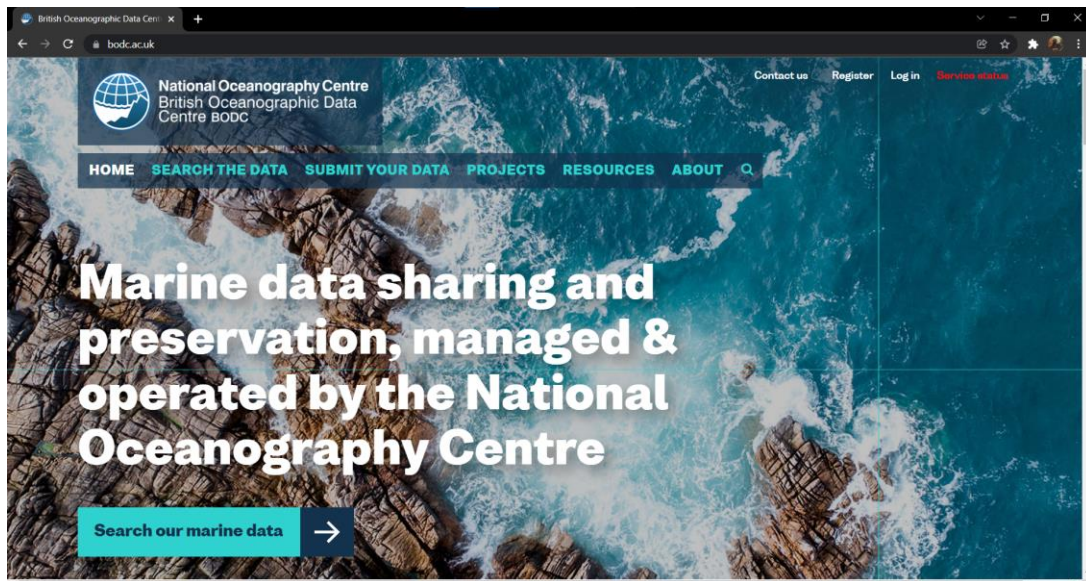


Berin Mašović 111-ST

Testiranje softvera Zadaća 3

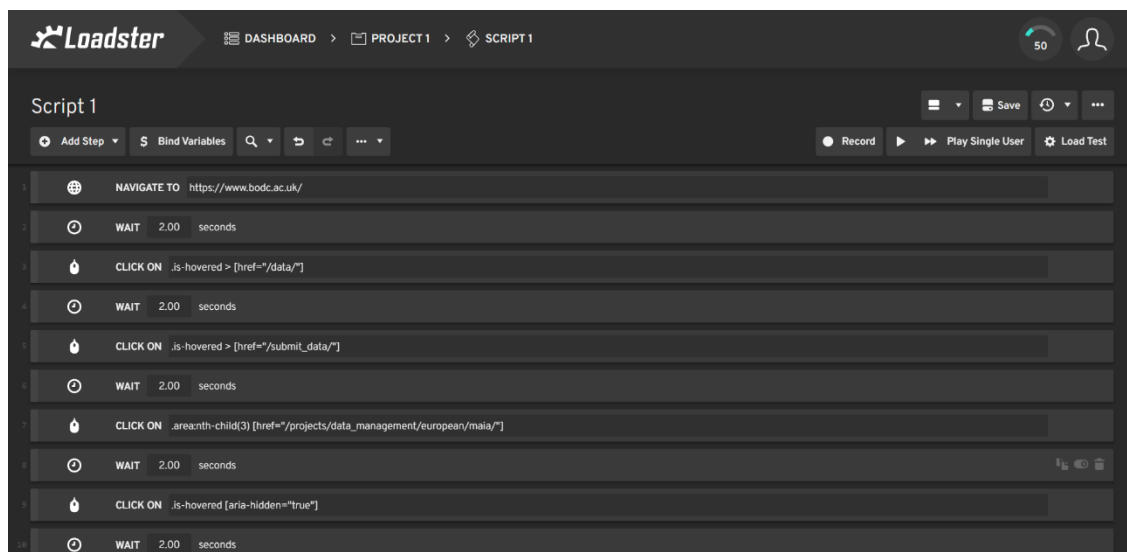
Zadatak 1.

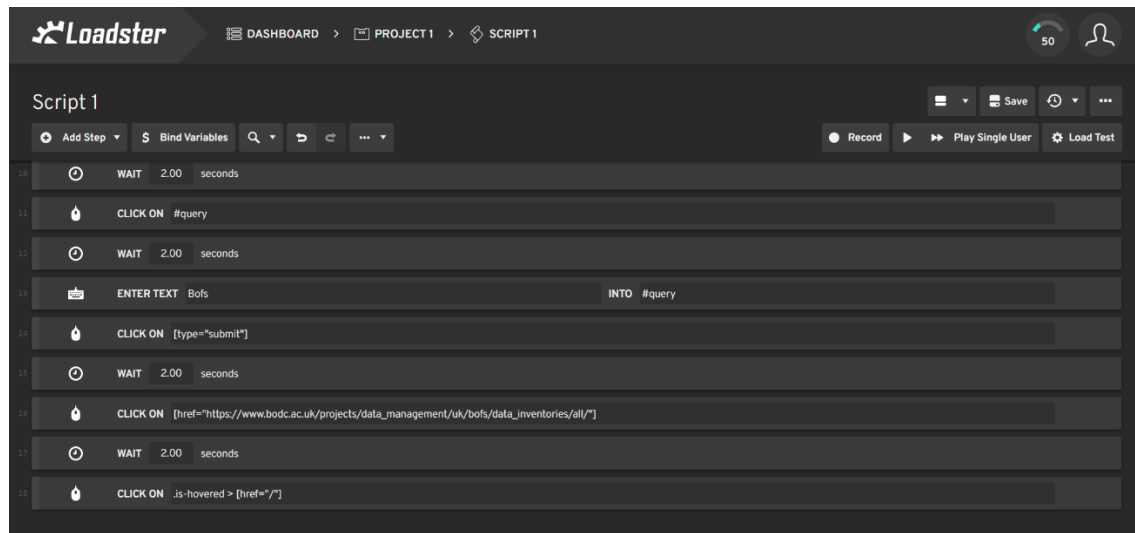
Stranicu koju sam dobio nakon slučajnog odabira je: <https://www.bodc.ac.uk/>



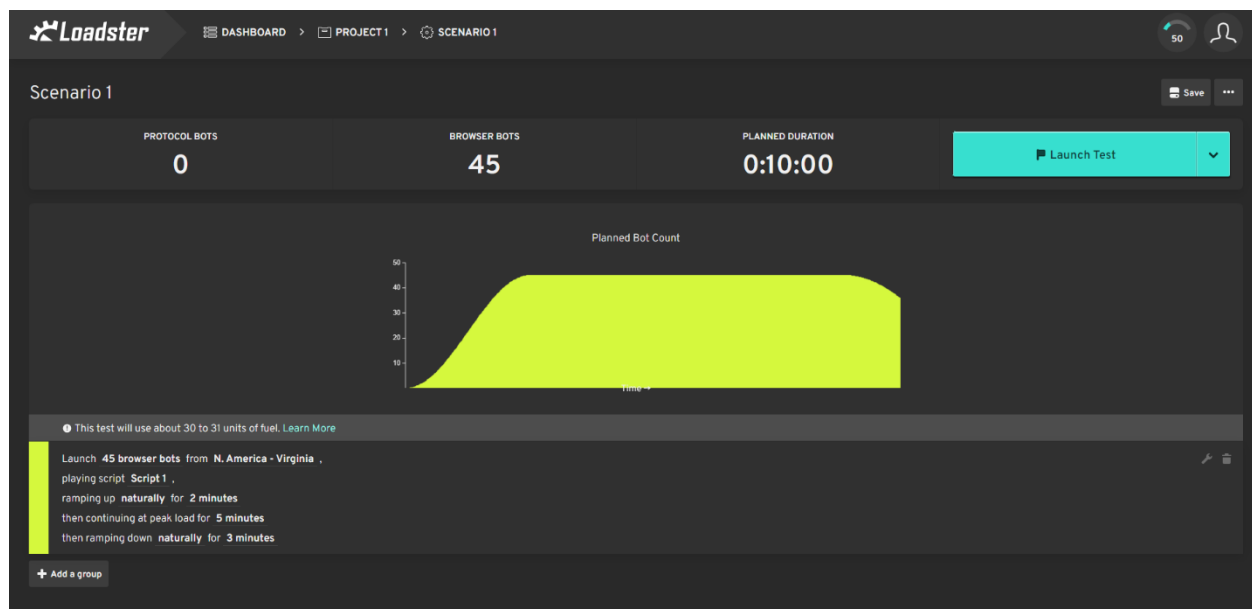
a.) Alat loadstar

Nakon kreiranja akaunta i prvog projekta dodajemo i testnu skriptu:

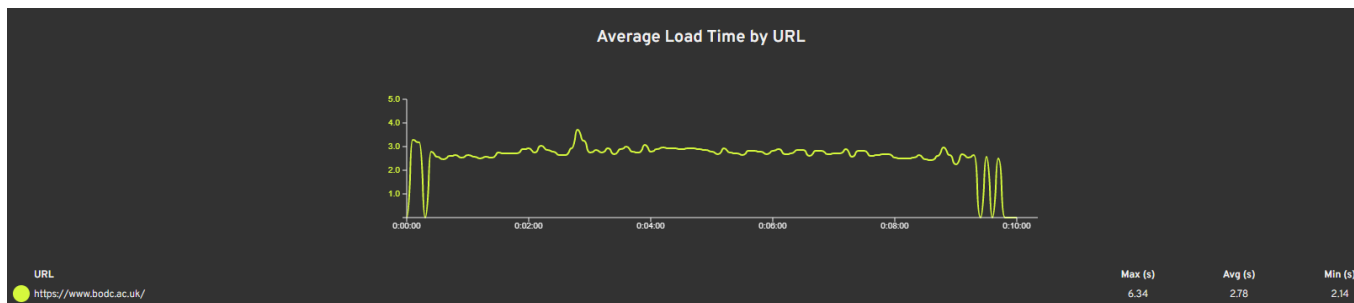




Sljedeći korak je kreiranje scenaria nakon čega pokrećemo testiranje:



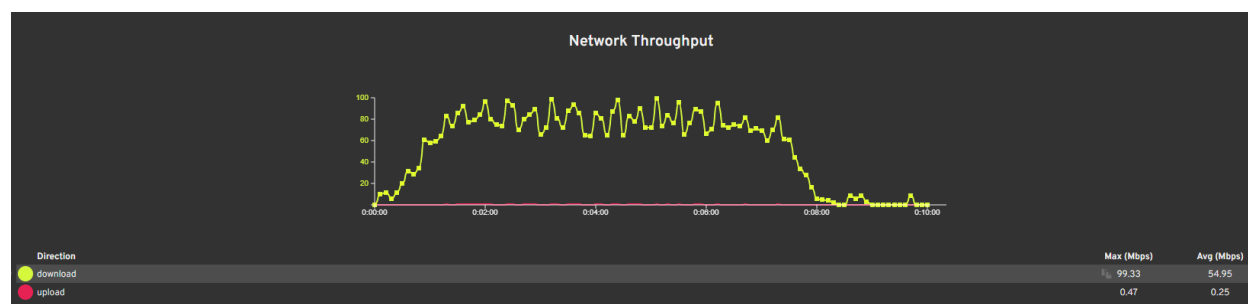
Pri završetku dobijamo kao rezultat mnogobrojne grafike od kojih je najvažniji graf srednjeg vremena učitavanja.



Sa grafa se može vidjeti kako se mijenjala količina vremena pristupa stranici kroz vrijeme i kroz povećanje broja virtuelnih korisnika. Maksimalno dostignuto vrijeme je 6,34s doke je minimalno 2,14s. Srednje vrijeme pristupa je 2,78s.

Sa grafa možemo uočiti da vrijeme pristupa nije linearno raslo sa rastom broja korisnika, kreće se na intervalu 2,6-3,1. Iz toga zaključujemo da je aplikacija u stanju da otprilike istom brzinom, za isto vrijeme pristupa opslužui zahtjeve bez obzira na broj korisnika.

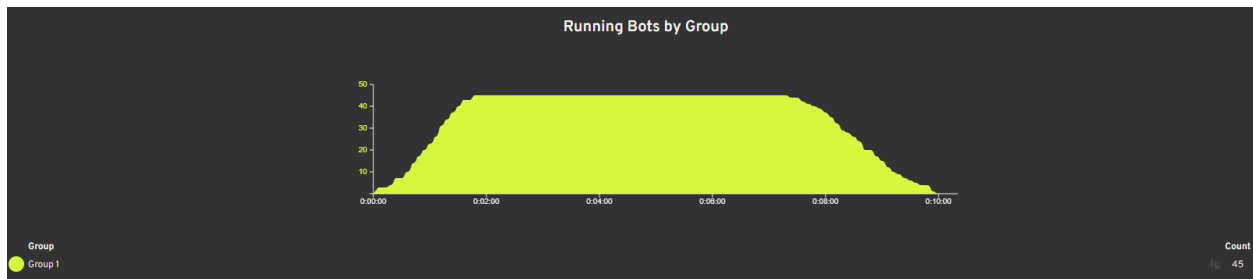
Sada će mo pogledati graf propusnosti mreže.



Sa ovog grafa uočavamo da je Max(Mbps) 99,33 a Min(Mbps) 54,95.

Primjećujemo da se sa porastom broja korisnika povećava i propusnost mreže no ne raste linearno već se održava na rasponu 80 – 100 Mbps.

Pogledat će mo još i sami tok izvršenja testa da bi smo provjerili da li se testiranje izvršilo na način kako smo zadali u samom scenariju.

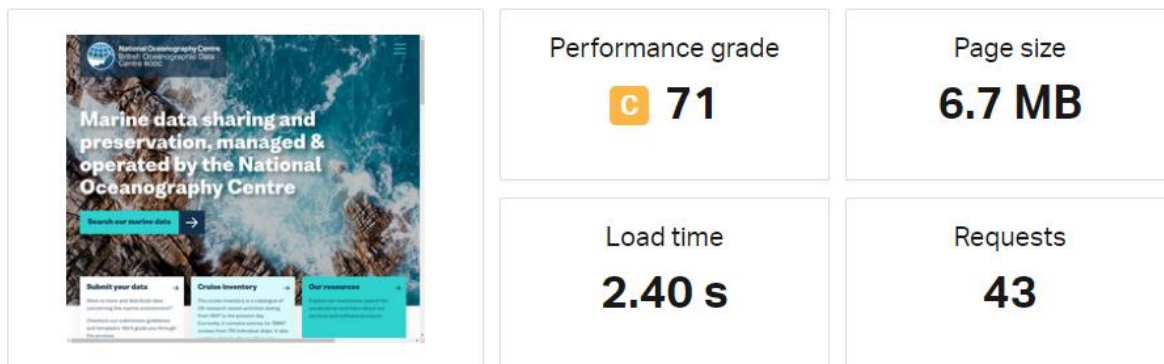


Sa grafa uočavamo da se testiranje izvršilo kako smo i zatražili. Broj virtuelnih korisnika je 45. Graf je skoro pa simetričan kao što smo i očekivali, rast broja korisnika se izvršavao 2 minute dok je pad nešto duže 3 minute.

b.) Alat Pingdom

Nakon unošenja URL stranice i pokretanja testa, dobijamo sljedeću metriku.

Your Results:



Podaci koje uočavamo su:

- Ukupna ocjena stranice: 71 – C
- Srednja vrijednost količine podataka potrebnih za učitavanje stranice: 6,7 MB
- Srednja vrijednost odziva na osnovu poslanih zahtjeva: 2.40s
- Broj poslanih zahtjeva na osnovu kojih su generisane vrijednosti metrika: 43

Iz ovih parametara možemo zaključiti da stranica i nije najbolje urađena, postoji dosta prostora za napredak. Pred njih postoje i mnoge druge informacije koje nam alat Pingdom prikazuje.

Improve page performance - prikazana su ocjene i savjeti kako poboljšati performance

Improve page performance

GRADE	SUGGESTION	
F 0	Add Expires headers	▼
F 1	Compress components with gzip	▼
F 36	Make fewer HTTP requests	▼
F 50	Use cookie-free domains	▲
When the browser requests a static image and sends cookies with the request, the server ignores the cookies. These cookies are unnecessary network traffic. To workaround this problem, make sure that static components are requested with cookie-free requests by creating a subdomain and hosting them there.		
C 75	Reduce DNS lookups	▼
A 100	Avoid empty src or href	▼
A 100	Put JavaScript at bottom	▼






Response codes – prikaz svih HTTP odgovora na poslane zahtjeve iz kojeg vidimo da su svi zahtjevi uspješno prošli sa statusom 200 - OK.

Response codes

RESPONSE CODE	RESPONSES
200 OK	43

Content size by content type – prikaz svih različitih tipova podataka potrebnih za učitavanje stranice, njihov procenat i veličina. Primjećujemo da se najviše koriste slike (87,34%).

Content size by content type

	PERCENT	
 Image	87.34%	5.8 MB
 Script	6.61%	440.1 KB
 Font	4.66%	310.2 KB
 HTML	0.56%	37.2 KB
 CSS	0.44%	29.0 KB
Total		

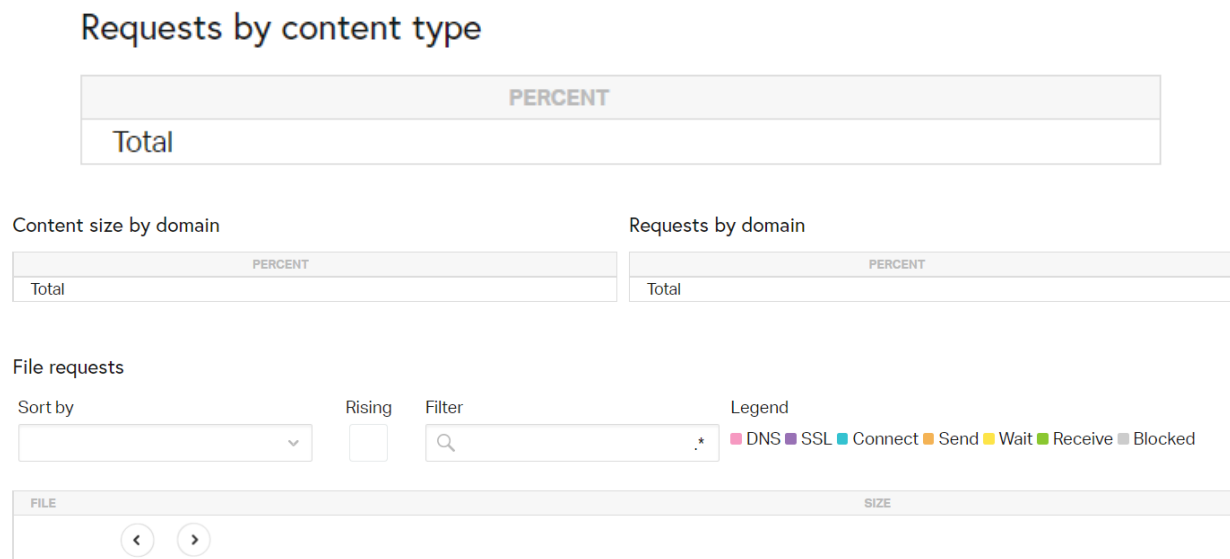
Postoje tu još mnoge drugih informacija koje nam alat nudi, no one za datu random odabranu stranicu nisu od koristi.

Requests by content type – prikaz svih podataka korištenih u poslanim zahtjevima

Content size by domain – prikaz svih domena korištenih u okviru stranice kojih u našem slučaju nije bilo

Requests by domain – prikaz domena korištenih u zahtjevima

File requests – hronološki prikaz mrežnih informacija na osnovu poslanih zahtjeva



c.) Kao dodatni alat ja sam izabrao alat LoadFocus (<https://loadfocus.com/>)

Create your first load test.

Select the Number of Virtual Users (concurrent users): ②

20

Execute Test

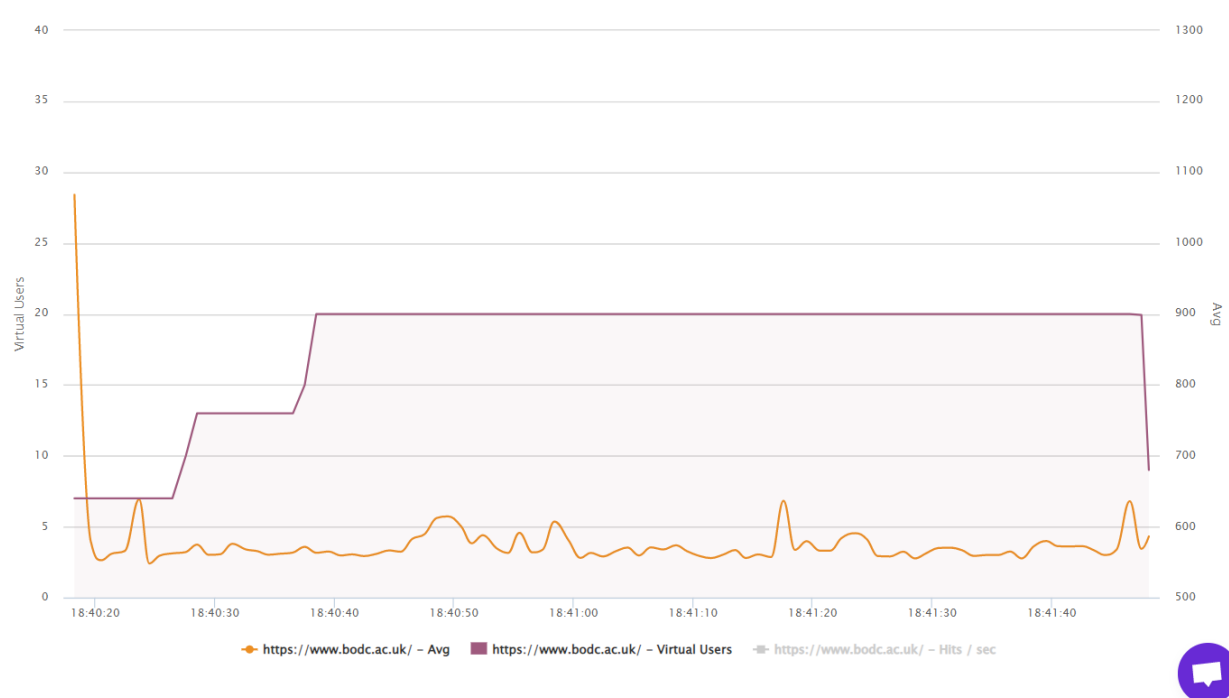
[Skip demo](#) [Go Back](#)

Nakon izvršenja dobijamo razne informacije.

Samples ?	Avg. Response Time ?	90% Response Time ?	Hits / sec ?	Errors % ?
2796 samples	572 ms	620 ms	31 hits/s	0 %

Neke od njih su prosječno vrijeme odziva: 572ms, broj uzoraka: 2796, broj grešaka: 0

Dostupan nam je i sljedeći graf.



Sa grafa možemo uočiti postepen rast broja virtuelnih korisnika i da su svi oni prestali s radom otprilike u isto vrijeme (ljubičasta krivulja).

Kada je u pitanju prosječno vrijeme izvršavanja krivulja je vrlo sličan onoj koju smo dobili testiranjem sa alatom loadstar.

Vrijeme pristupa ne raste linearno sa porastom broja korisnika. Zaključujemo da je aplikacija u stanju da otprilike istom brzinom, za isto vrijeme pristupa opsluži zahtjeve bez obzira o kolikom broju korisnika se radilo.

Zadatak 2.

U projektu implementiranom u zadaći 1 i 2 postoji switch-case koji se ponavlja 3 puta. Korišten je za odabir veličine niza odnosno Liste intova, stingova i Osoba.

Njegova kompleksnost iznosi 5 (po jedan za svaki case), s obzirom da imamo 3 takva switch-casea ukupna kompleksnost je 15.

```
int vel = 0;

switch (odabirVel)
{
    case 1:
        vel = 10;
        break;

    case 2:
        vel = 100;
        break;

    case 3:
        vel = 1000;
        break;

    case 4:
        Console.Write("\nUnesite broj elemenata niza:");
        vel = Convert.ToInt32(Console.ReadLine());
        break;

    default:
        Console.WriteLine("Pogrešan unos");
        return;
}
```



```

int vel2 = 0;

switch (odabirVel2)
{
    case 1:
        vel2 = 10;
        break;

    case 2:
        vel2 = 100;
        break;

    case 3:
        vel2 = 1000;
        break;

    case 4:
        Console.Write("\nUnesite broj elemenata niza:");
        vel2 = Convert.ToInt32(Console.ReadLine());
        break;

    default:
        Console.WriteLine("Pogrešan unos");
        return;
}

```

```

int vel3 = 0;

switch (odabirVel3)
{
    case 1:
        vel3 = 10;
        break;

    case 2:
        vel3 = 100;
        break;

    case 3:
        vel3 = 1000;
        break;

    case 4:
        Console.Write("\nUnesite broj elemenata niza:");
        vel3 = Convert.ToInt32(Console.ReadLine());
        break;

    default:
        Console.WriteLine("Pogrešan unos");
        return;
}

```

Da se uvjerimo da je kompleksnost zaista 15. Provjeriti ćemo kompleksnost i kroz VisualStudio.

Code Metrics Results								
Filter:	Min:	Max:						
Hierarchy	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Source code	Lines of Executable code		
▾ Zadaca2 (Debug)	63	228	1	22	1,412	307		
▾ Zadaca2	63	228	1	22	1,412	307		
▾ RandomDateTime	65	1	1	4	11	6		
▾ StubInterface	100	5	0	0	10	0		
▾ ZamjenskiObjekat	81	5	1	1	28	10		
▾ AlgoritamBubbleSort	65	11	1	2	46	13		
▾ AlgoritamGnomeSort	62	11	1	2	52	17		
▾ AlgoritamInsertionSort	63	11	1	4	45	15		
▾ AlgoritamSelectionSort	60	13	1	2	57	19		
▾ AlgoritamShellSort	59	13	1	2	49	21		
▾ AlgoritamCombSort	61	15	1	2	68	28		
▾ AlgoritamOddEvenSort	54	15	1	2	77	31		
▾ AlgoritamCocktailSort	50	17	1	2	98	43		
▾ AlgoritamHeapSort	61	21	1	2	90	35		
▾ Osoba	82	25	1	1	96	28		
▾ Program	28	65	1	18	671	41		
▾ Main(string[]) : void	28	65		18	668	41		

Vidimo da kompleksnost za Program iznosi 65. Ako sada izbrišemo data 3 switch-casea kompleksnost Programa iznosi 50. Ovim je dokazano da je kompleksnost za data 3 switch-casea zaista 15, tj kompleksnost pojedinačnog switch-casea je 5.

Code Metrics Results								
Filter:	Min:	Max:						
Hierarchy	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Source code	Lines of Executable code		
▾ Zadaca2 (Debug)	63	213	1	23	1,332	307		
▾ Zadaca2	63	213	1	23	1,332	307		
▾ RandomDateTime	65	1	1	4	11	6		
▾ StubInterface	100	5	0	0	10	0		
▾ ZamjenskiObjekat	81	5	1	1	28	10		
▾ AlgoritamBubbleSort	65	11	1	2	46	13		
▾ AlgoritamGnomeSort	62	11	1	2	52	17		
▾ AlgoritamInsertionSort	63	11	1	4	45	15		
▾ AlgoritamSelectionSort	60	13	1	2	57	19		
▾ AlgoritamShellSort	59	13	1	2	49	21		
▾ AlgoritamCombSort	61	15	1	2	68	28		
▾ AlgoritamOddEvenSort	54	15	1	2	77	31		
▾ AlgoritamCocktailSort	50	17	1	2	98	43		
▾ AlgoritamHeapSort	61	21	1	2	90	35		
▾ Osoba	82	25	1	1	96	28		
▾ Program	30	50	1	19	591	41		
▾ Main(string[]) : void	30	50		19	588	41		

Da bi smanjili ciklomatsku kompleksnost vršimo refaktoring. Switch-case je potrebno pretvoriti u funkciju Odaberi.

```

3 references
private static int Odabir(int n)
{
    int[] velicina = { 10, 100, 1000 };

    if (n >= 1 && n <= 3)
    {
        return velicina[n - 1];
    }
    else if (n == 4)
    {
        Console.WriteLine("\nUnesite broj elemenata niza:");
        int vel = Convert.ToInt32(Console.ReadLine());
        return vel;
    }
    Console.WriteLine("Pogrešan unos");
    System.Environment.Exit(0);
    return 0;
}

```

Poziv funkcije će izgledati:

```
int vel = Odabir(odabirVel);
```

Kompleksnost funkcije je 4.

2 za if jer postoje 2 uslova unutar njega ($n \geq 1 \ \&\& \ n \leq 3$), 1 za else if i 1 za samu deklaraciju funkcije.

Ovo ćemo provjeriti kroz Visual Studio.

Code Metrics Results							
Filter:	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Source code	Lines of Executable code	
<div> <div> <div></div> <div>Zadaca2 (Debug)</div> </div> <div> <div></div> <div>Zadaca2</div> </div> <div> <div></div> <div>RandomDateTime</div> </div> <div> <div></div> <div>StubInterface</div> </div> <div> <div></div> <div>ZamjenskiObjekat</div> </div> <div> <div></div> <div>AlgoritamBubbleSort</div> </div> <div> <div></div> <div>AlgoritamGnomeSort</div> </div> <div> <div></div> <div>AlgoritamInsertionSort</div> </div> <div> <div></div> <div>AlgoritamSelectionSort</div> </div> <div> <div></div> <div>AlgoritamShellSort</div> </div> <div> <div></div> <div>AlgoritamCombSort</div> </div> <div> <div></div> <div>AlgoritamOddEvenSort</div> </div> <div> <div></div> <div>AlgoritamCocktailSort</div> </div> <div> <div></div> <div>AlgoritamHeapSort</div> </div> <div> <div></div> <div>Osoba</div> </div> <div> <div></div> <div>Program</div> </div> <div> <div></div> <div>Main(string[]): void</div> </div> <div> <div></div> <div>Odabir(int): int</div> </div> </div>							
	64	217	1	23	1,360	317	
	64	217	1	23	1,360	317	
	65	1	1	4	11	6	
	100	5	0	0	10	0	
	81	5	1	1	28	10	
	65	11	1	2	46	13	
	62	11	1	2	52	17	
	63	11	1	4	45	15	
	60	13	1	2	57	19	
	59	13	1	2	49	21	
	61	15	1	2	68	28	
	54	15	1	2	77	31	
	50	17	1	2	98	43	
	61	21	1	2	90	35	
	82	25	1	1	96	28	
	40	54	1	19	619	51	
	30	50		18	596	41	
	62	4		3	20	10	

Vidimo da je kompleksnost funkcije Odabir zaista 4. Refaktoringom smo uspjeli da ciklomatsku kompleksnost Programa umanjimo za 11, bila je 65 a sada iznosi 54. To smo postigli tako što smo 3 switch-casea kompleksnosti 15 zamijenili sa funkcijom kompleksnosti 4.